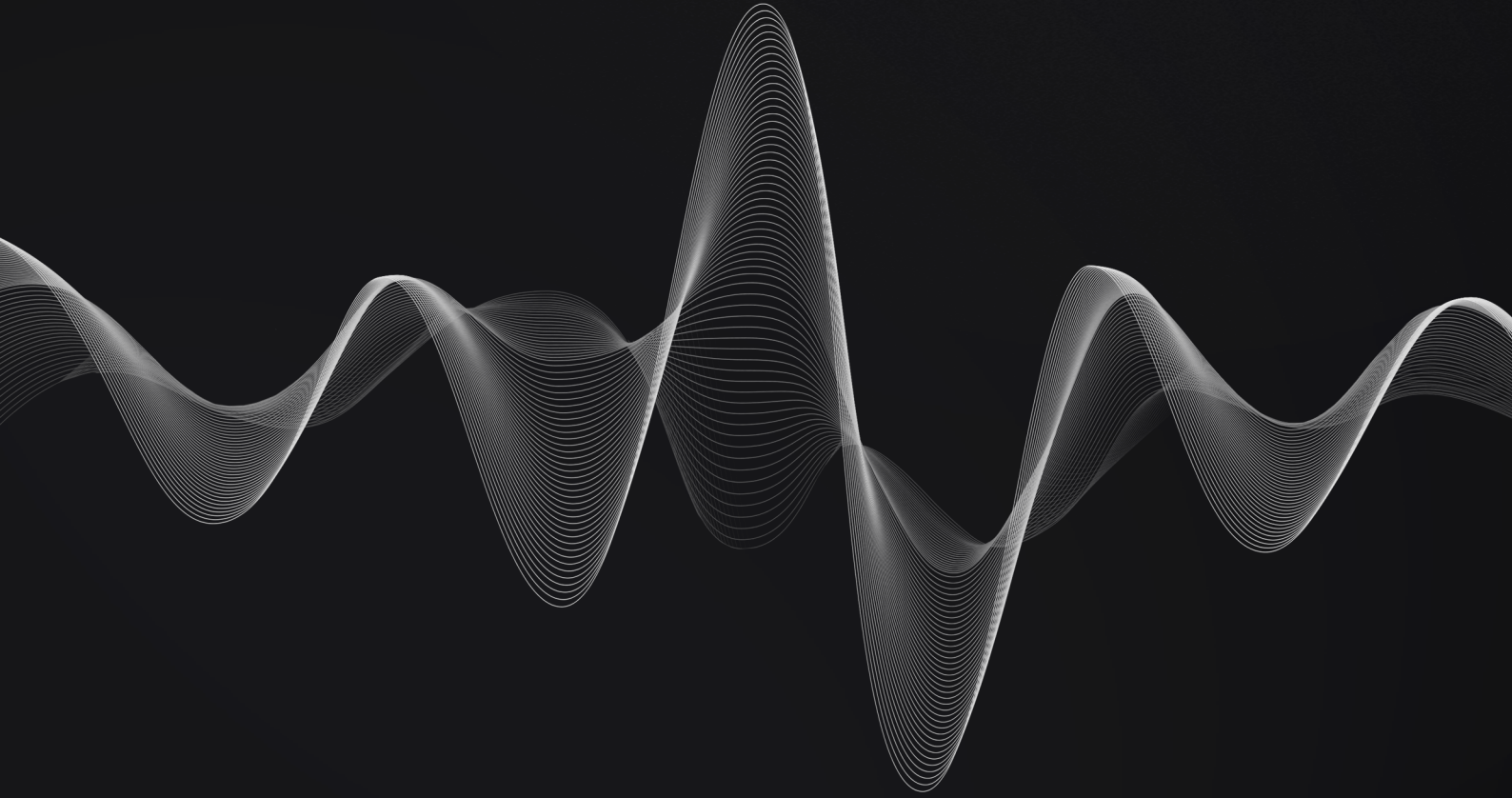


# **Nodle**

## Nodle Rollup Bridge Audit Report







# Document Control

**PUBLIC**

**FINAL**(v2.0)

## Audit\_Report\_NODL-RBR\_FINAL\_20

Apr 29, 2024		v0.1	João Simões: Initial draft
May 2, 2024		v0.2	Michał Bajor: Added findings
May 3, 2024		v0.3	Michał Bazyli: Added findings
May 3, 2024		v1.0	Charles Dray: Approved
May 9, 2024		v1.1	João Simões: Reviewed findings
May 10, 2024		v2.0	Charles Dray: Published

<b>Points of Contact</b>	Elliott	Teisson-	Nodle	eliott@nodle.com
	niere			
	Charles Dray		Resonance	charles@resonance.security
<b>Testing Team</b>	João Simões		Resonance	joao.simoes@resonance.security
	Ilan Abitbol		Resonance	ilan.abitbol@resonance.security
	Michał Bazyli		Resonance	michal.bazyli@resonance.security
	Michał Bajor		Resonance	michal.bajor@resonance.security

## Copyright and Disclaimer

© 2024 Resonance Security, Inc. All rights reserved.

The information in this report is considered confidential and proprietary by Resonance and is licensed to the recipient solely under the terms of the project statement of work. Reproduction or distribution, in whole or in part, is strictly prohibited without the express written permission of Resonance.

All activities performed by Resonance in connection with this project were carried out in accordance with the project statement of work and agreed-upon project plan. It's important to note that security assessments are time-limited and may depend on information provided by the client, its affiliates, or partners. As such, the findings documented in this report should not be considered a comprehensive list of all security issues, flaws, or defects in the target system or codebase.

Furthermore, it is hereby assumed that all of the risks in electing not to remedy the security issues identified henceforth are sole responsibility of the respective client. The acknowledgement and understanding of the risks which may arise due to failure to remedy the described security issues, waives and releases any claims against Resonance, now known or hereafter known, on account of damage or financial loss.

# Contents

<b>1 Document Control</b>	<b>2</b>
Copyright and Disclaimer .....	2
<b>2 Executive Summary</b>	<b>4</b>
System Overview .....	4
Repository Coverage and Quality.....	4
<b>3 Target</b>	<b>5</b>
<b>4 Methodology</b>	<b>6</b>
Severity Rating.....	7
Repository Coverage and Quality Rating.....	8
<b>5 Findings</b>	<b>9</b>
Insecure Handling Of Sensitive Information In Command-Line Arguments .....	10
Potential Double-Spending Due To Possible Block Reverts .....	11
Floating Pragma .....	12
Lack Of EIP1191 Check When Initializing Bridging Operation .....	13
Lack Of min_amount Update Function .....	14
<b>A Proof of Concepts</b>	<b>15</b>

# Executive Summary

**Nodle** contracted the services of Resonance to conduct a comprehensive security audit of their smart contracts between April 26, 2024 and May 3, 2024. The primary objective of the assessment was to identify any potential security vulnerabilities and ensure the correct functioning of smart contract operations.

During the engagement, Resonance allocated 3 engineers to perform the security review. The engineers, including an accomplished professional with extensive proficiency in blockchain and smart-contract security, encompassing specialized skills in advanced penetration testing, and in-depth knowledge of multiple blockchain protocols, devoted 5 days to the project. The project's test targets, overview, and coverage details are available throughout the next sections of the report.

The ultimate goal of the audit was to provide Nodle with a detailed summary of the findings, including any identified vulnerabilities, and recommendations to mitigate any discovered risks. The results of the audit are presented in detail further below.

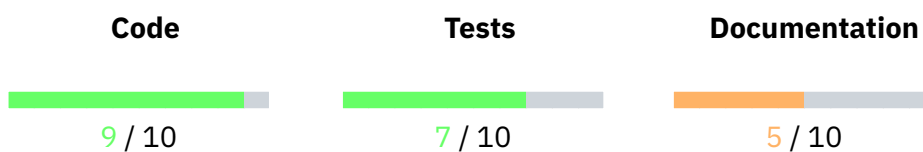


## System Overview

The Noodle Rollup Bridge is a bridge to transfers asset exclusively from the Node parachain to zkSync, leveraging a Rust-written Oracle, an Ink Smart Contract, and a zkSync Smart Contract.



## Repository Coverage and Quality



Resonance's testing team has assessed the Code, Tests, and Documentation coverage and quality of the system and achieved the following results:

- The code follows development best practices and makes use of known patterns, standard libraries, and language guides. It is easily readable and uses the latest stable version of relevant components. Overall, **code quality is excellent**.
- Unit tests are included. The tests cover both technical and functional requirements. Overall, **tests coverage and quality is good**.
- The documentation only includes the specification of the system and relevant explanations of workflows and interactions. Overall, **documentation coverage and quality is average**.

# Target

The objective of this project is to conduct a comprehensive review and security analysis of the smart contracts that are contained within the specified repository.

The following items are included as targets of the security assessment:

- Repository: [NodleCode/rollup/src](#)
- Hash: 3a12cb55591a78d86f8baa6aff9707bee7e266cb
- Repository: [NodleCode/bridge](#)
- Hash: b847107cf83e50802372e5bfa99baf82a5c34ef5

The following items are excluded:

- External and standard libraries
- Files pertaining to the deployment process
- Financial-related attack vectors

# Methodology

In the context of security audits, Resonance's primary objective is to portray the workflow of a real-world cyber attack against an entity or organization, and document in a report the findings, vulnerabilities, and techniques used by malicious actors. While several approaches can be taken into consideration during the assessment, Resonance's core value comes from the ability to correlate automated and manual analysis of system components and reach a comprehensive understanding and awareness with the customer on security-related issues.

Resonance implements several and extensive verifications based off industry's standards, such as, identification and exploitation of security vulnerabilities both public and proprietary, static and dynamic testing of relevant workflows, adherence and knowledge of security best practices, assurance of system specifications and requirements, and more. Resonance's approach is therefore consistent, credible and essential, for customers to maintain a low degree of risk exposure.

Ultimately, product owners are able to analyze the audit from the perspective of a malicious actor and distinguish where, how, and why security gaps exist in their assets, and mitigate them in a timely fashion.

## Source Code Review - Solidity EVM

During source code reviews for Web3 assets, Resonance includes a specific methodology that better attempts to effectively test the system in check:

1. Review specifications, documentation, and functionalities
2. Assert functionalities work as intended and specified
3. Deploy system in test environment and execute deployment processes and tests
4. Perform automated code review with public and proprietary tools
5. Perform manual code review with several experienced engineers
6. Attempt to discover and exploit security-related findings
7. Examine code quality and adherence to development and security best practices
8. Specify concise recommendations and action items
9. Revise mitigating efforts and validate the security of the system

Additionally and specifically for Solidity EVM audits, the following attack scenarios and tests are recreated by Resonance to guarantee the most thorough coverage of the codebase:

- Reentrancy attacks
- Frontrunning attacks
- Unsafe external calls
- Unsafe third party integrations
- Denial of service
- Access control issues

- Inaccurate business logic implementations
- Incorrect gas usage
- Arithmetic issues
- Unsafe callbacks
- Timestamp dependence
- Mishandled panics, errors and exceptions

## Severity Rating

Security findings identified by Resonance are rated based on a Severity Rating which is, in turn, calculated off the **impact** and **likelihood** of a related security incident taking place. This rating provides a way to capture the principal characteristics of a finding in these two categories and produce a score reflecting its severity. The score can then be translated into a qualitative representation to help customers properly assess and prioritize their vulnerability management processes.

The **impact** of a finding can be categorized in the following levels:

1. Weak - Inconsequential or minimal damage or loss
2. Medium - Temporary or partial damage or loss
3. Strong - Significant or unrecoverable damage or loss

The **likelihood** of a finding can be categorized in the following levels:

1. Unlikely - Requires substantial knowledge or effort or uncontrollable conditions
2. Likely - Requires technical knowledge or no special conditions
3. Very Likely - Requires trivial knowledge or effort or no conditions

		Likelihood		
		Very Likely	Likely	Unlikely
Impact	Strong	Critical	High	Medium
	Medium	High	Medium	Low
	Weak	Medium	Low	Info



# Repository Coverage and Quality Rating

The assessment of Code, Tests, and Documentation coverage and quality is one of many goals of Resonance to maintain a high-level of accountability and excellence in building the Web3 industry. In Resonance it is believed to be paramount that builders start off with a good supporting base, not only development-wise, but also with the different security aspects in mind. A product, well thought out and built right from the start, is inherently a more secure product, and has the potential to be a game-changer for Web3's new generation of blockchains, smart contracts, and dApps.

Accordingly, Resonance implements the evaluation of the code, the tests, and the documentation on a score **from 1 to 10** (1 being the lowest and 10 being the highest) to assess their quality and coverage. In more detail:

- Code should follow development best practices, including usage of known patterns, standard libraries, and language guides. It should be easily readable throughout its structure, completed with relevant comments, and make use of the latest stable version components, which most of the times are naturally more secure.
- Tests should always be included to assess both technical and functional requirements of the system. Unit testing alone does not provide sufficient knowledge about the correct functioning of the code. Integration tests are often where most security issues are found, and should always be included. Furthermore, the tests should cover the entirety of the codebase, making sure no line of code is left unchecked.
- Documentation should provide sufficient knowledge for the users of the system. It is useful for developers and power-users to understand the technical and specification details behind each section of the code, as well as, regular users who need to discern the different functional workflows to interact with the system.



# Findings

During the security audit, several findings were identified to possess a certain degree of security-related weaknesses. These findings, represented by unique IDs, are detailed in this section with relevant information including Severity, Category, Status, Code Section, Description, and Recommendation. Further extensive information may be included in corresponding appendices should it be required.

An overview of all the identified findings is outlined in the table below, where they are sorted by Severity and include a **Remediation Priority** metric asserted by Resonance's Testing Team. This metric characterizes findings as follows:

- ||||| "Quick Win" Requires little work for a high impact on risk reduction.
- |||| "Standard Fix" Requires an average amount of work to fully reduce the risk.
- ||| "Heavy Project" Requires extensive work for a low impact on risk reduction.

---

Findings ID	Description	Severity	Status
RES-01	Insecure Handling Of Sensitive Information In Command-Line Arguments		Resolved
RES-02	Potential Double-Spending Due To Possible Block Reverts		Resolved
RES-03	Floating Pragma		Resolved
RES-04	Lack Of EIP1191 Check When Initializing Bridging Operation		Resolved
RES-05	Lack Of min_amount Update Function		Acknowledged



# Insecure Handling Of Sensitive Information In Command-Line Arguments

Medium RES-NODL-RBR01

Sensitive Data Exposure

Resolved

## Code Section

- `oracle/src/main.rs#L114`

## Description

During the security assessment of the oracle binary, it was observed that sensitive information, specifically private keys, are required to be passed as command-line arguments when executing the binary. This practice exposes private keys in the process's command line, which can be accessed by any user on the same system with permissions to view the process list, as well as potentially being exposed in logging or in other monitoring tools accessible on the Google Cloud platform. This represents a significant security risk as command-line arguments are not a secure medium for transmitting sensitive information due to their high visibility.

## Recommendation

It is recommended to:

- **Use Environment Variables:** Modify the application to accept sensitive data such as private keys through environment variables rather than command-line arguments. Environment variables can be secured and scoped at the process level, reducing the risk of exposure.
- **Leverage Google Cloud's Secret Management:** Utilize Google Cloud's Secret Manager to handle sensitive configurations securely. Secret Manager stores, manages, and accesses secrets such as API keys and credentials, with built-in encryption, ensuring that sensitive information is not exposed to unauthorized users or through inadvertent logging.
- **Implement Secure Configuration Management:** Adopt a secure configuration management process that includes regular reviews and updates to ensure that sensitive data is handled securely throughout the application lifecycle. This process should also include the use of secure storage mechanisms for sensitive information and proper access controls.
- **Audit and Logging:** Ensure that logging mechanisms mask or exclude sensitive information. Audit logs should be generated for access to sensitive data, and logs should be regularly reviewed to detect unauthorized access attempts.
- **Security Awareness and Training:** Conduct regular security training for developers and operations teams on best practices for handling sensitive information and the potential risks associated with insecure practices like passing sensitive data in command-line arguments.

## Status

*This issue was resolved in commit 8a5f34dfac1d61aef6ff1ff5850532d28537e4e3.*



# Potential Double-Spending Due To Possible Block Reverts

Medium RES-NODL-RBR02

Architecture

Resolved

## Code Section

- `oracle/src/main.rs`

## Description

The off-chain component of the bridge runs periodically and processes some number of blocks. The off-chain components are sending transactions to the target network as they are processing the blocks. It was observed that there is no explicit check assuring that the block being processed is finalized. As a consequence, in a scenario when an unfinalized block is reverted, however it was already processed by the off-chain component, a double-spending opportunity arises. User whose interaction with the `evm_bridge` contract was reverted can do it again in new block which would result in a second valid bridging operation from the off-chain component's perspective.

## Recommendation

It is recommended to implement an explicit check on the blocks being processed that would assure they are finalized and block reverts are not possible.

## Status

*This issue was resolved in commit `02818893fe9acbfe4b1f38c9338134056a2cdadb`.*



# Floating Pragma

Info

RES-NODL-RBR03

Code Quality

Resolved

## Code Section

Not specified.

## Description

Floating pragmas is a feature of Solidity that allows developers to specify a range of compiler versions that can be used to compile the smart contract code. For security purposes specifically, the usage of floating pragmas is discouraged and not recommended. Contracts should be compiled and deployed with a strict pragma, the one which was thoroughly tested the most by developers. Locking the pragma helps ensure that contracts do not accidentally get deployed using too old or too recent compiler versions that might introduce security issues that impact the contract negatively.

It should be noted however that, pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package.

The smart contracts of Nodle protocol make use of floating pragmas. These are not intended for use as libraries for other developers and, as such, should be locked of their pragma.

## Recommendation

It is recommended to choose a specific compiler version and use it for every contract in the project.

## Status

*This issue was resolved in commit [df2cc4eafb46bd377dcd6f24828ecdc327728fda](#).*



# Lack Of EIP1191 Check When Initializing Bridging Operation

Info

RES-NODL-RBR04

Data Validation

Resolved

## Code Section

- [parachain-contract/lib.rs#L11](#)

## Description

The `evm_bridge` contract parses the Ethereum-compatible address to make sure that it is a legitimate one and bridging process can be performed successfully. However, it was observed that the check does not take into account the `chainId` values, i.e. it does not verify them in accordance with EIP1191 standard. It does not impact the security of the contract, however it is considered a best practice, because it will limit user error.

## Recommendation

It is recommended to enable the checksum check with the `chainId` to leverage the EIP1191 standard.

## Status

*This issue was resolved in commit [8f149ebe8828aacb863efc6efbe4b8d47b03ec78](#).*



# Lack Of min\_amount Update Function

Info

RES-NODL-RBR05

Business Logic

Acknowledged

## Code Section

- `parachain-contract/lib.rs`

## Description

The `evm_bridge` contract stores a `min_amount` variable. This variable dictates the minimal amount of transferred value that needs to be attached to initiate the bridging operation. This variable is there to make sure only the transfers with enough value are processed so that costs related to the subsequent off-chain and on-chain processing are reasonable. However, the situation might change in the future which may result in lowering or increasing the costs of processing. The `evm_bridge` contract does not contain a function that would allow to modify the `min_amount` value. Hence, if the costs related to the processing would increase, it might result in some of the bridging operations being unreasonable from the cost-effect perspective.

## Recommendation

It is recommended to introduce a setter function that would allow changing the `min_amount` value. This function should be callable only by a privileged user.

## Status

*The issue was acknowledged by Nodle team. The development team stated "We expressively chose to not include any potential upgradeability or admin functions within the contracts. If necessary we can easily redeploy the same contract with slightly differing parameters instead and update minting permissions on the NODL contract."*

# Proof of Concepts

*No Proof-of-Concept was deemed relevant to describe findings in this engagement.*